On the Underestimation of Model Uncertainty by Bayesian K-nearest Neighbors

Wanhua Su[1], Hugh Chipman[2,*], Mu Zhu[1]

[1] Department of Statistics and Actuarial Science,
University of Waterloo, Waterloo, Ontario, Canada N2L 3G1
[2] Department of Mathematics and Statistics,
Acadia University, Wolfville, Nova Scotia, Canada B4P 2R6

April 8, 2008

Abstract

When using the K-nearest neighbors method, one often ignores uncertainty in the choice of K. To account for such uncertainty, Holmes and Adams (2002) proposed a Bayesian framework for K-nearest neighbors (KNN). Their Bayesian KNN (BKNN) approach uses a pseudo-likelihood function, and standard Markov chain Monte Carlo (MCMC) techniques to draw posterior samples. Holmes and Adams (2002) focused on the performance of BKNN in terms of misclassification error but did not assess its ability to quantify uncertainty. We present some evidence to show that BKNN still significantly underestimates model uncertainty.

Keywords: bootstrap interval; MCMC; posterior interval; pseudo-likelihood.

1 Introduction

The K-nearest neighbors method (e.g., Fix and Hodges 1951; Cover and Hart 1967) is conceptually simple but flexible and useful in practice. It can be used for both regression and classification. We focus on classification only.

Under the assumption that points close to one another should have similar responses, KNN classifies a new observation according to the class labels of its K nearest neighbors. In order to identify the neighbors, one must decide how to measure proximity among points and how to define the neighborhood. The most commonly-used distance metric is the Euclidean distance. The tuning parameter, K, is normally chosen by cross-validation. Figure 1 illustrates how KNN works. Suppose one takes $K = 5$. The possible predicted values are $\{0/5, 1/5, \cdots, 5/5\}$. Among those five nearest neighbors of test point A, four out of five belong to class 0. Therefore, A is classified to class 0 with an estimated probability of 4/5. Similarly, test point B is classified to class 1 with an estimated probability of 4/5.

*corresponding author; email: `hugh dot chipman at acadiau dot ca`

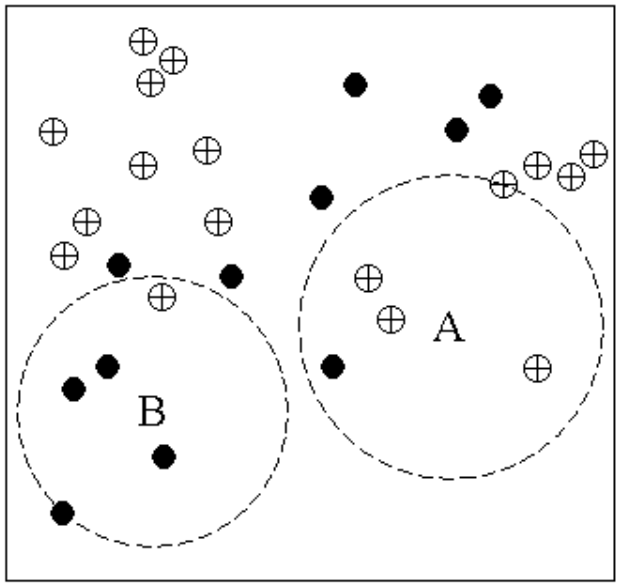

Figure 1: Simulated example illustrating KNN with $K = 5$. Training observations from class 0 are indicated by the symbol "$\oplus$", and those from class 1 are indicated by the symbol "$\bullet$". A and B are two test points.

Holmes and Adams (2002) pointed out that regular KNN does not account for the uncertainty in the choice of K. They presented a Bayesian framework for KNN (BKNN), compared its performance with the regular KNN on several benchmark data sets and concluded that BKNN outperformed KNN in terms of misclassification error. By model averaging over the posterior of K, BKNN is able to improve predictive performance. Unfortunately, they never assessed the inferential aspect of BKNN. In this paper, we present some evidence to show that, even though BKNN is designed to capture the uncertainty in the choice of K, it still significantly underestimates overall uncertainty.

2 BKNN

We first give a quick overview of BKNN in the context of a classification problem with Q different classes. To cast KNN into a Bayesian framework, Holmes and Adams (2002) adopted the following (pseudo) likelihood function for the data:

$$p(\mathbf{Y}|\mathbf{X}, \beta, K) = \prod_{i=1}^{n} p(y_i|\mathbf{x}_i, \beta, K) = \prod_{i=1}^{n} \frac{\exp\{(\beta/K)\sum_{j\in N(\mathbf{x}_i,K)} \mathrm{I}(y_j = y_i)\}}{\sum_{q=1}^{Q} \exp\{(\beta/K)\sum_{j\in N(\mathbf{x}_i,K)} \mathrm{I}(y_j = q)\}}. \quad (1)$$

The indicator function I is 1 whenever its argument is true, and the notation "$j \in N(\mathbf{x}_i, K)$" identifies the indices j of the K-nearest neighbours of $\mathbf{x}_i$. Thus $\sum_{j\in N(\mathbf{x}_i,K)} \mathrm{I}(y_j = y_i)$ is K times the estimated probability from a conventional KNN model.

There are two unknown parameters, K and β. The parameter K is an positive integer controlling the number of nearest neighbors; and β is a positive continuous parameter governing the strength of interaction between a data point and its neighbors.

The likelihood function (1) is a so-called pseudo-likelihood function (see, e.g., Besag 1974, 1975). Unlike regular likelihood functions, the component for data point y_i depends on the class labels of other data points y_j, for $j \neq i$. Treating β and K as random variables, the marginal predictive distribution for a new data point $(\mathbf{x}_{n+1}, y_{n+1})$ based on the training data $(\mathbf{X}, \mathbf{Y})$ is given by

$$p(y_{n+1}|\mathbf{x}_{n+1}, \mathbf{X}, \mathbf{Y}) = \sum_K \int p(y_{n+1}|\mathbf{x}_{n+1}, \mathbf{X}, \mathbf{Y}, \beta, K) p(\beta, K|\mathbf{X}, \mathbf{Y}) d\beta, \tag{2}$$

where

$$p(\beta, K|\mathbf{X}, \mathbf{Y}) \propto p(\mathbf{Y}|\mathbf{X}, \beta, K) p(\beta, K)$$

is the posterior distribution of (β, K).

Except for the fact that β should be positive, little prior knowledge is known on the likely values of K and β. Therefore, Holmes and Adams (2002) adopt independent, non-informative prior densities,

$$p(\beta, K) = p(\beta)p(K)$$

where

$$p(K) = \text{UNIF}[1, \ldots, K_{\max}] \text{ with } K_{\max} = n, \quad p(\beta) = c\text{I}(\beta > 0),$$

and c is a constant so that $p(\beta)$ is an improper flat prior on $\mathbb{R}^+$.

A random-walk Metropolis-Hastings algorithm is then used to draw M samples from the posterior $p(\beta, K|\mathbf{X}, \mathbf{Y})$, so that (2) can be approximated by

$$p(y_{n+1}|\mathbf{x}_{n+1}, \mathbf{X}, \mathbf{Y}) \approx \frac{1}{M} \sum_{j=1}^{M} p(y_{n+1}|\mathbf{x}_{n+1}, \mathbf{X}, \mathbf{Y}, \beta^{(j)}, K^{(j)}), \tag{3}$$

where $(K^{(j)}, \beta^{(j)})$ is the jth sample from the posterior.

3 Experiments and results

One might believe that the Bayesian formulation will automatically account for model uncertainty, and that this is a major advantage of BKNN over regular KNN. We now describe a simple experiment that shows BKNN still significantly underestimates model uncertainty.

The same experiment is repeated 100 times. Each time, we first generate $n = 250$ pairs of training data from a known, two-class model (details in Section 3.1). We then fit BKNN and regular KNN on the training data, and let them make predictions at a set of 160 pre-selected test points (details in Section 3.2). For each test point, say $(\mathbf{x}_{n+1}, y_{n+1})$, our parameter of interest is

$$\theta_{n+1} \equiv \Pr(y_{n+1} = 1|\mathbf{x}_{n+1}). \tag{4}$$

We construct both point estimates (Section 3.3) and interval estimates (Section 3.4) of θ_{n+1}: $\hat{\theta}_{n+1}$ and $\hat{I}_{n+1}$.

To fit BKNN, we use the Matlab code provided by Holmes and Adams (2002) and exactly the same MCMC setting as described in Holmes and Adams (2002, Section 3.1). To fit regular KNN, we use the `knn` function in R.

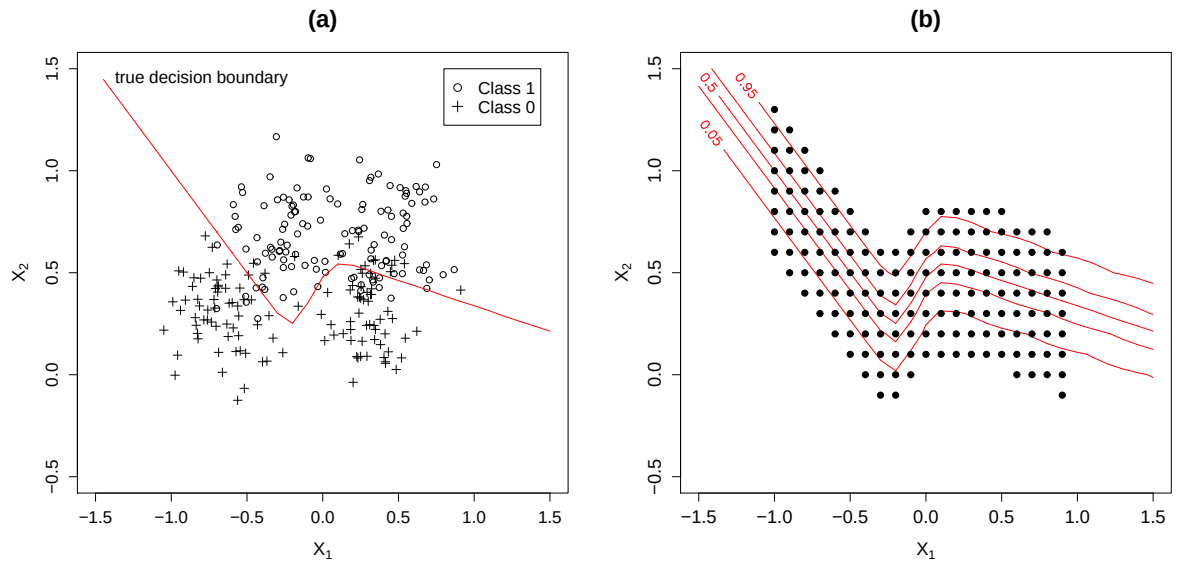

Figure 2: (a) Training data from one experiment, and the true probability contour, $\Pr(y = 1|\mathbf{x})$, as given by (5). (b) The fixed set of test points, and the true probability contour.

3.1 Simulation Model

Holmes and Adams (2002, Section 3.1) illustrated BKNN with a synthetic dataset consisting of 250 training and 1000 test points, taken from `http://www.stats.ox.ac.uk/pub/PRNN`. These data were originally generated from two classes, each being an equal mixture of two bivariate normal distributions. In order to be able to generate slightly different training data every time we repeat our experiment, we imitate this synthetic data set by assuming the underlying distributions of class 1 (C_1) and class 0 (C_0) to be:

$$\begin{aligned}\mathbf{x}|C_1 \sim f_1(\mathbf{x}) &= 0.5\text{BVN}\left(\boldsymbol{\mu}_{11}, \boldsymbol{\Sigma}\right) + 0.5\text{BVN}\left(\boldsymbol{\mu}_{12}, \boldsymbol{\Sigma}\right) \\ \mathbf{x}|C_0 \sim f_0(\mathbf{x}) &= 0.5\text{BVN}\left(\boldsymbol{\mu}_{01}, \boldsymbol{\Sigma}\right) + 0.5\text{BVN}\left(\boldsymbol{\mu}_{02}, \boldsymbol{\Sigma}\right),\end{aligned}$$

with

$$\boldsymbol{\mu}_{11} = \begin{pmatrix} -0.3 \\ 0.7 \end{pmatrix}, \quad \boldsymbol{\mu}_{12} = \begin{pmatrix} 0.4 \\ 0.7 \end{pmatrix}, \quad \boldsymbol{\mu}_{01} = \begin{pmatrix} -0.7 \\ 0.3 \end{pmatrix}, \quad \boldsymbol{\mu}_{02} = \begin{pmatrix} 0.3 \\ 0.3 \end{pmatrix}$$

and

$$\boldsymbol{\Sigma} = \begin{pmatrix} 0.03 & 0 \\ 0 & 0.03 \end{pmatrix}.$$

The prior class probabilities are taken to be equal, i.e., $\Pr(y = 1) = \Pr(y = 0) = 0.5$. Given any data point $\mathbf{x}$, its posterior probability of being in C_1 can be calculated by Bayes' rule

$$\Pr(y = 1|\mathbf{x}) = \frac{0.5 f_1(\mathbf{x})}{0.5 f_1(\mathbf{x}) + 0.5 f_0(\mathbf{x})}. \tag{5}$$

Figure 2(a) shows the training data from one experiment and the true decision boundary.

3.2 Test points

Instead of focusing on the total misclassification error, we focus on predictions made at a *fixed* set of test points. These test points are chosen as follows: first, we lay out a grid along the first coordinate, $X_1 \in \{-1, -0.9, -0.8, \cdots, 0.8, 0.9\}$; for each X_1 in that grid, eight different values of X_2 are chosen so that the test points together "cover" the critical part of the true posterior probability contour. A total of 160 test points are obtained this way. Figure 2(b) shows the fixed set of test points and the true posterior probability contour, $\Pr(y = 1|\mathbf{x})$, as given by (5).

In what follows, we refer to θ_{n+1} as the key parameter of interest, but it should be understood that the subscript "$n+1$" is used to refer to any test point. There are altogether 160 such test points, and exactly the same calculations are performed for all of them, not just one of them.

3.3 Point estimates of θ_{n+1}

For BKNN, the point estimate of $\theta_{n+1} \equiv \Pr(y_{n+1} = 1|\mathbf{x}_{n+1})$ is the posterior mean:

$$\hat{\theta}_{n+1}^{BKNN} = \frac{1}{M} \sum_{j=1}^{M} \Pr(y_{n+1} = 1|\mathbf{x}_{n+1}, \mathbf{X}, \mathbf{Y}, \beta^{(j)}, K^{(j)}),$$

where $(K^{(j)}, \beta^{(j)})$ are samples drawn from the posterior distribution, $p(K, \beta|\mathbf{X}, \mathbf{Y})$. For regular KNN, one chooses the parameter K by cross-validation, and normally uses the original KNN score

$$\tilde{\theta}_{n+1}^{KNN} = \frac{1}{K} \sum_{j \in N(\mathbf{x}_{n+1}, K)} \mathrm{I}(y_j = 1)$$

as the point estimate. In order to make things fully comparable, however, we further transform the KNN scores by a logistic model fitted using the training data. We describe this next.

Notice that, for binary classification problems, i.e., $Q = 2$, each multiplicative term in (1) can be rewritten as

$$\begin{aligned} p(y_i|\mathbf{x}_i, \beta, K) &= \frac{\exp\{(\beta/K) \sum_{j \in N(\mathbf{x}_i, K)} \mathrm{I}(y_j = y_i)\}}{\exp\{(\beta/K) \sum_{j \in N(\mathbf{x}_i, K)} \mathrm{I}(y_j = y_i)\} + \exp\{(\beta/K) \sum_{j \in N(\mathbf{x}_i, K)} \mathrm{I}(y_j \neq y_i)\}} \\ &\overset{(\dagger)}{=} \frac{\exp\{(\beta/K)[2 \sum_{j \in N(\mathbf{x}_i, K)} \mathrm{I}(y_j = y_i) - K]\}}{1 + \exp\{(\beta/K)[2 \sum_{j \in N(\mathbf{x}_i, K)} \mathrm{I}(y_j = y_i) - K]\}} \\ &= \frac{\exp\{\beta[2g(y_i) - 1]\}}{1 + \exp\{\beta[2g(y_i) - 1]\}}, \end{aligned} \tag{6}$$

where

$$g(y_i) = \frac{1}{K} \sum_{j \in N(\mathbf{x}_i, K)} \mathrm{I}(y_j = y_i), \tag{7}$$

is the output of KNN. The step labelled (†) in (6) is due to the identity

$$\sum_{j \in N(\mathbf{x}_i, K)} I(y_j = y_i) + \sum_{j \in N(\mathbf{x}_i, K)} I(y_j \neq y_i) = K.$$

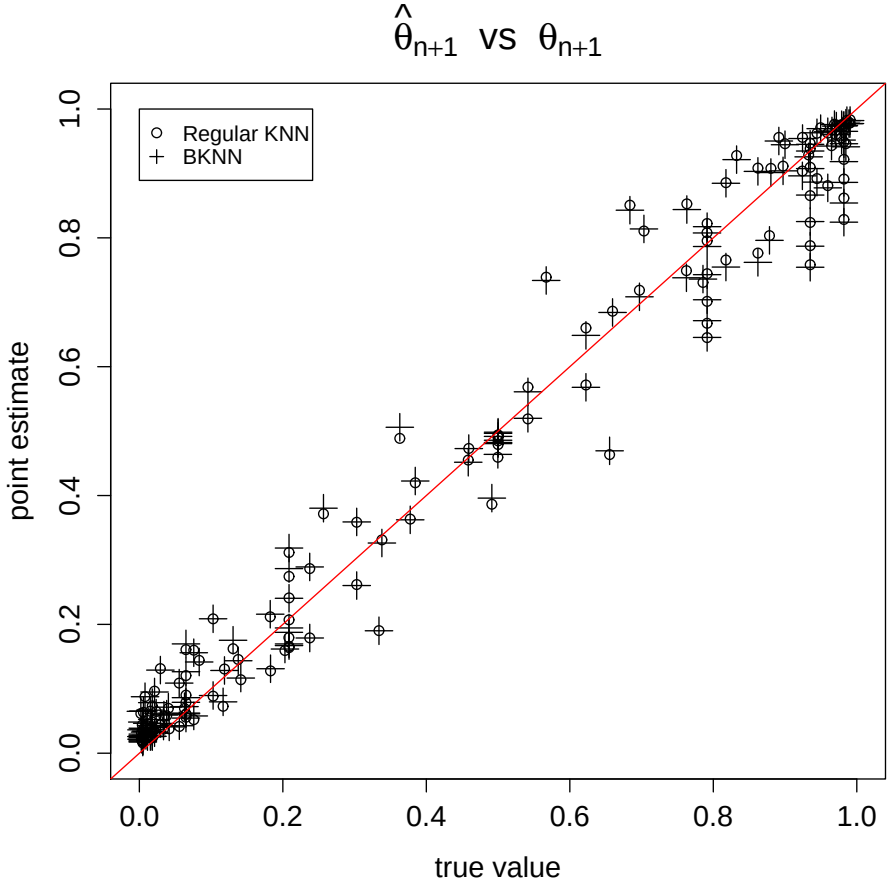

Figure 3: Average of 100 point estimates versus the true parameter value, for all 160 test points. A 45-degree reference line going through the origin is also displayed.

Notice that (6) is equivalent to running a logistic regression with no intercept and $[2g(y_i)-1]$ as the only covariate. Since this extra transformation is built into BKNN, we use

$$\hat{\theta}_{n+1}^{KNN} = \frac{\exp\{\hat{\beta}[2\tilde{\theta}_{n+1}^{KNN} - 1]\}}{1 + \exp\{\hat{\beta}[2\tilde{\theta}_{n+1}^{KNN} - 1]\}} \tag{8}$$

as the point estimate of regular KNN in order to be fully comparable with BKNN. In (8), $\hat{\beta}$ is obtained by running a logistic regression of y_i onto $[2g(y_i) - 1]$ with no intercept using the training data.

After repeating the experiment 100 times, we obtain 100 slightly different point estimates at each $\mathbf{x}_{n+1}$. Figure 3 plots the average of these 100 point estimates against the true value for all 160 test points. We see that both BKNN and regular KNN give very similar point estimates.

3.4 Interval estimates of θ_{n+1}

The main focus of our experiments is interval estimation. In particular, we are interested in the question of whether these interval estimates adequately capture model uncertainty.

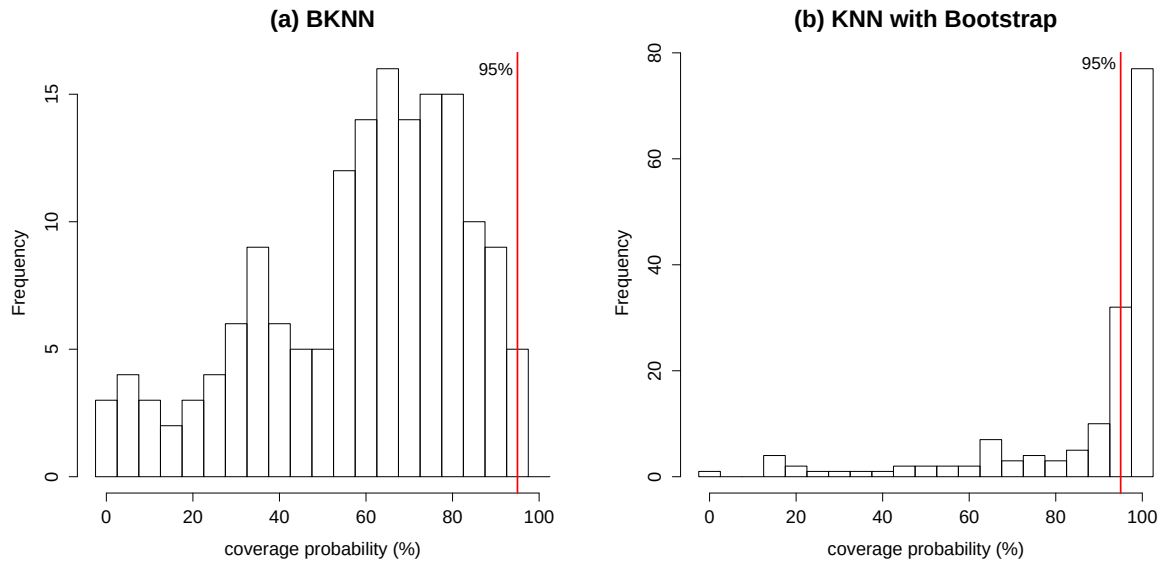

Figure 4: Estimated coverage probabilities of (a) $\hat{I}_{n+1}^{BKNN}$ and (b) $\hat{I}_{n+1}^{KNN}$, for all 160 test points.

For BKNN, we use the 95% posterior (or credible) interval as our interval estimate, $\hat{I}_{n+1}^{BKNN}$. This is constructed by finding the 2.5th and 97.5th percentiles of the posterior samples. To obtain an interval estimate for regular KNN, $\hat{I}_{n+1}^{KNN}$, we resort to Efron's bootstrap. Given a training set, $\mathcal{D}$, we generate 500 bootstrap samples, $\mathcal{D}_1^*, \mathcal{D}_2^*, \cdots, \mathcal{D}_{500}^*$, and repeat the entire KNN model building process — that is, choosing K by cross-validation and calculating $\hat{\theta}_{n+1,b}$ according to (8) — for every $\mathcal{D}_b^*$, $b = 1, 2, \cdots, 500$. The interval estimate of θ_{n+1} is constructed by taking the 2.5th and 97.5th percentiles of the set, $\{\hat{\theta}_{n+1,1}, \cdots, \hat{\theta}_{n+1,500}\}$.

3.4.1 Coverage probabilities

Our first question of interest is: What are the coverage probabilities of $\hat{I}_{n+1}^{KNN}$ and $\hat{I}_{n+1}^{BKNN}$? After repeating the experiment 100 times, we obtain 100 slightly different interval estimates at each $\mathbf{x}_{n+1}$. The coverage probability of $\hat{I}_{n+1}^{BKNN}$ (and that of $\hat{I}_{n+1}^{KNN}$) can be estimated easily by counting the number of times θ_{n+1} is included in the interval over the 100 experiments. Histograms of the estimated coverage probabilities for all 160 test points are shown in Figure 4. The posterior intervals produced by BKNN can easily be seen to have fairly poor coverage overall.

3.4.2 Lengths

For each interval estimate, we can also calculate its length, e.g.,

$$\begin{aligned} \text{length}_{n+1}^{BKNN} &= \left| \hat{\theta}_{n+1}^{BKNN,97.5\%} - \hat{\theta}_{n+1}^{BKNN,2.5\%} \right|, \\ \text{length}_{n+1}^{KNN} &= \left| \hat{\theta}_{n+1}^{KNN,97.5\%} - \hat{\theta}_{n+1}^{KNN,2.5\%} \right|. \end{aligned}$$

Let

$$\overline{\text{length}}_{n+1}^{BKNN} \quad \text{and} \quad \overline{\text{length}}_{n+1}^{KNN}$$

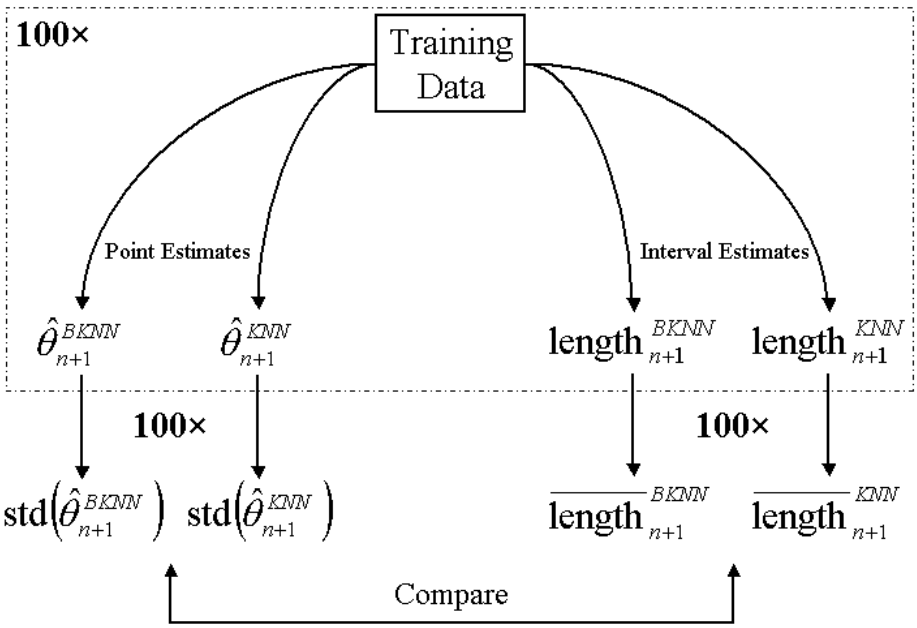

Figure 5: Schematic illustration of our assessment protocol. Variation over 100 point estimates is used as a benchmark to assess the quality of the corresponding interval estimates.

be the average lengths of these 100 interval estimates. Our second question of interest is: Are they too long, too short, or just right? In order to answer this question, we need a "gold standard".

The very reason for using these interval estimates is to reflect that there is uncertainty in our estimate of the underlying parameter, θ_{n+1}. This uncertainty is easy to assess directly when one can repeatedly generate different sets of training data and repeatedly estimate the parameter, which is exactly what we have done. The standard deviations of the 100 point estimates (Section 3.3), which we write as

$$\text{std}(\hat{\theta}_{n+1}^{BKNN}) \quad \text{and} \quad \text{std}(\hat{\theta}_{n+1}^{KNN}),$$

give us a direct assessment of this uncertainty.

If the point estimates, $\hat{\theta}_{n+1}^{BKNN}$ and $\hat{\theta}_{n+1}^{KNN}$, are approximately normally distributed, then the correct lengths of the corresponding interval estimates should be roughly 4 times the aforementioned standard deviation, that is,

$$\overline{\text{length}}_{n+1}^{BKNN} \approx 4 \times \text{std}(\hat{\theta}_{n+1}^{BKNN}), \tag{9}$$

$$\overline{\text{length}}_{n+1}^{KNN} \approx 4 \times \text{std}(\hat{\theta}_{n+1}^{KNN}). \tag{10}$$

We use (9)-(10) as *heuristic* guidelines to assess how well the interval estimates can capture model uncertainty, despite lack of formal justification for the normal approximation. Figure 5 provides a schematic illustration of our assessment protocol.

Figure 6 plots the average lengths of these 100 interval estimates against 4 times the standard deviations of the corresponding point estimates — that is, $\overline{\text{length}}_{n+1}^{BKNN}$ against

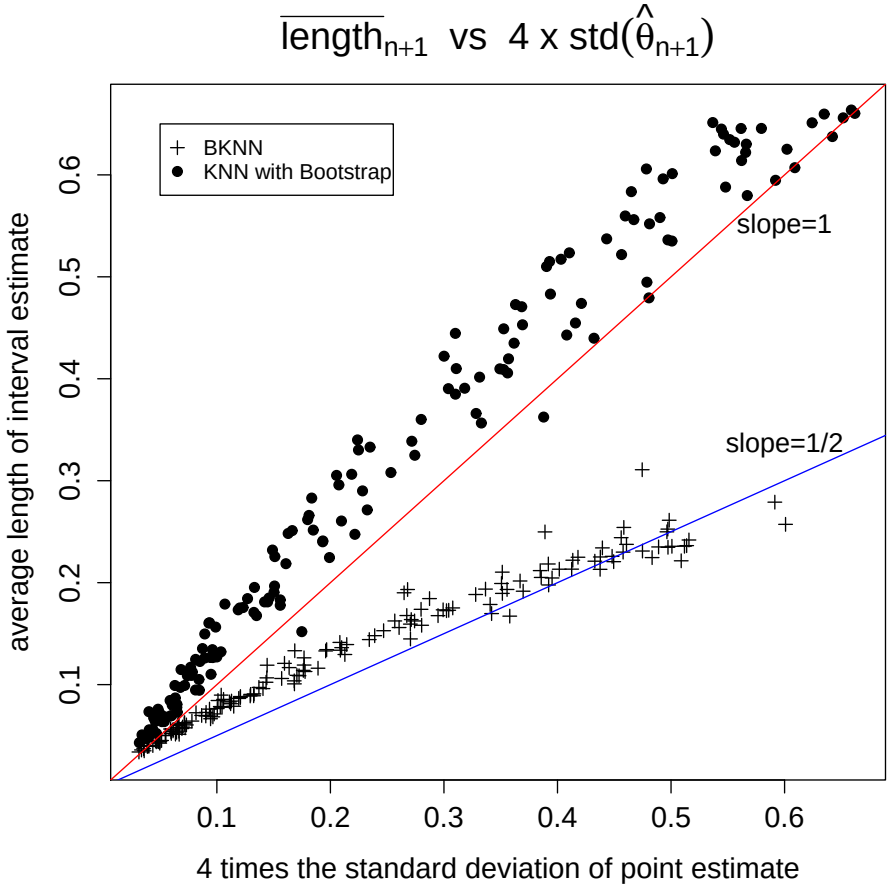

Figure 6: Average length of 100 interval estimates versus 4 times the standard deviation of the corresponding point estimate, for all 160 test points. Two reference lines – both going through the origin, one with slope=1 and another with slope=1/2 — are also displayed.

$4 \times \text{std}(\hat{\theta}_{n+1}^{BKNN})$ and $\overline{\text{length}}_{n+1}^{KNN}$ against $4 \times \text{std}(\hat{\theta}_{n+1}^{KNN})$ — for all 160 test points. Here, it is easy to see that the Bayesian posterior intervals are apparently too short, whereas bootstrapping regular KNN gives a more accurate assessment of the amount of uncertainty in the point estimate.

4 Discussion

Why does BKNN underestimate uncertainty? We believe it is because BKNN only accounts for the uncertainty in the *number* of neighbors (i.e., the parameter K), but it is unable to account for the uncertainty in the spatial *locations* of these neighbors. This is a general phenomenon associated with pseudo-likelihood functions.

Pseudo-likelihood functions were first introduced by Besag (1974, 1975) to model spatial interactions in lattice systems. Since then, they have been widely used in image processing (e.g., Besag 1986) and network tomography (e.g., Strauss and Ikeda 1990; Liang and Yu 2003; Robins *et al.* 2007). However, statistical inference based on pseudo-likelihood func-

tions is still in its infancy. Some researchers argue that pseudo-likelihood inference can be problematic since it ignores at least part of the dependence structure in the data. In applications to model social networks, a number of researchers, such as Wasserman and Robins (2005) and Snijders (2002), have pointed out that maximum pseudo-likelihood estimates are substantially biased and the standard errors of the parameters are generally underestimated. For BKNN, the pseudo-likelihood function (1) clearly ignores the fact that the locations of one's neighbors are also random, not just the number of neighbors.

However, for complex networks whose full likelihood functions are intractable, models based on pseudo-likelihood are attractive (if not the only) alternatives (Strauss and Ikeda 1990). Rather than trying to write down the full likelihood functions for these difficult problems, it is probably more fruitful to concentrate our research efforts on how to adjust or correct standard error estimates produced by the pseudo-likelihood. To this effect, one interesting observation from Figure 6 is the fact that

$$\overline{\text{length}}_{n+1}^{BKNN} \approx 2 \times \text{std}(\hat{\theta}_{n+1}^{BKNN}).$$

If we continue to use $4 \times \text{std}(\hat{\theta}_{n+1}^{BKNN})$ as the "gold standard", then these Bayesian posterior intervals are about half as long as they should be. We have observed this phenomenon on other examples, too, but do not yet have an explanation for it.

Despite the fact that BKNN seems to underestimate overall uncertainty, that $\overline{\text{length}}_{n+1}^{BKNN}$ is still approximately proportional to $\text{std}(\hat{\theta}_{n+1}^{BKNN})$ suggests that we can still rely on it to assess the *relative* uncertainty of its predictions. For many practical problems, this is still very useful. For example, if two accounts, A and B, are both predicted to be fraudulent with a high probability of 0.9 but the posterior interval of A is twice as long as that of B, then it is natural for a financial institution to spend its limited resources investigating account B rather than account A.

Acknowledgment

This research is partially supported by the Natural Science and Engineering Research Council (NSERC) of Canada, Canada's National Institute for Complex Data Structures (NICDS) and the Mathematics of Information Technology And Complex Systems (MITACS) network.

References

Besag, J. (1974). Spatial interaction and the statistical analysis of lattice systems (with discussion). *Journal of Royal Statistical Society: Series B*, **36**(2), 192–236.

Besag, J. (1975). Statistical analysis of non-lattice data. *The Statistician*, **24**(3), 179–195.

Besag, J. (1986). On the statistical analysis of dirty pictures. *Journal of Royal Statistical Society: Series B*, **48**(3), 259–302.

Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, **IT-13**, 21–27.

Fix, E. and Hodges, J. L. (1951). Discriminatory analysis–nonparametric discrimination: Consistency properties. Technical report, USAF School of Aviation Medicine, Randolph Field, Texas.

Holmes, C. C. and Adams, N. M. (2002). A probabilistic nearest neighbour method for statistical pattern recognition. *Journal of Royal Statistical Society: Series B*, **64**(2), 295–306.

Liang, G. and Yu, B. (2003). Maximum pseudo likelihood estimation in network tomography. *IEEE Transactions on Signal Processing*, **51**, 2043–2053.

Robins, G., Pattison, P., Kalish, Y., and Lusher, D. (2007). An introduction to exponential random graph (P^*) models for social networks. *Social Networks*, **29**, 173–191.

Snijders, T. A. B. (2002). Markov chain monte carlo estimation of exponential random graph model. *Journal of Social Structure*, **3**(2).

Strauss, D. and Ikeda, M. (1990). Pseudolikelihood estimation for social networks. *Journal of the American Statistical Association*, **85**, 204–212.

Wasserman, S. S. and Robins, G. L. (2005). An introduction to random graphs, dependence graphs, and p^*. In J. S. P. Carrrington and S. S. Wasserman, editors, *Models and Methods in Social Network Analysis*, pages 148–161. Cambridge University Press, Cambridge.